

A $(2 + \epsilon)$ -Approximation for Maximum Weight Matching in the Semi-Streaming Model*

Ami Paz[§]

Gregory Schwartzman[§]

February 16, 2017

Abstract

We present a simple deterministic single-pass $(2 + \epsilon)$ -approximation algorithm for the maximum weight matching problem in the semi-streaming model. This improves upon the currently best known approximation ratio of $(3.5 + \epsilon)$.

Our algorithm uses $O(n \log^2 n)$ space for constant values of ϵ . It relies on a variation of the local-ratio theorem, which may be of use for other algorithms in the semi-streaming model as well.

1 Introduction

We present a simple $(2 + \epsilon)$ -approximation algorithm for the maximum weight matching (MWM) problem in the semi-streaming model. Our algorithm is deterministic, single-pass, requires only $O(1)$ processing time per incoming edge, and uses $O(n \log^2 n)$ space for any constant $\epsilon > 0$. This improves upon the previously best known approximation algorithm of Crouch and Stubbs [CS14, GMZ16], which achieves an approximation ratio of $(3.5 + \epsilon)$ and takes $O(\log n)$ time to process an edge.

The MWM problem is a classical problem in graph theory. Its first efficient solution is due to Edmonds [Edm65], which was later improved by Micali and Vazirani [MV80]. The MWM problem was one of the first to be considered in the semi-streaming model when this model was first presented [FKM⁺05], and probably the most studied problem in this model since (see “Related Work”).

In the first algorithms for the MWM problem in the semi-streaming model, a matching is maintained at all times, and is being updated according to the incoming edges. More recent algorithms sort the edges into weight classes, keep a subset of each class, and then find a matching in the union of these subsets.

Like previous algorithms, our algorithm maintains a set of edges from which the final matching is constructed; however, we do not maintain a matching at all times, but only construct it in a post-processing stage. Our main technical contribution is the adaptation of the local-ratio technique for maximization problems [BYE85, BBF⁺01] to the semi-streaming model, in a novel, yet simple,

*An extended abstract of this work was presented in SODA 2017 [PS17].

[§]Department of Computer Science, Technion, Israel. {amipaz,gregorys}@cs.technion.ac.il
Supported in part by ISF grant 1696/14.

manner. Our work presents a significantly better approximation ratio for the MWM problem, along with a new approximation technique for optimization problems in the semi-streaming model.

For the maximum *unweighted* matching problem, a simple greedy algorithm yields a 2-approximation. This was observed in the very first paper on the semi-streaming model [FKM⁺05], and not improved since. Any future improvement of our result by more than an ϵ -factor will also solve this long-standing problem.

Our Contribution When developing an algorithmic framework for a new model, it is natural to first address the most fundamental algorithmic problems. Finding a large matching in a graph is indeed a fundamental problem, which has been extensively studied in the model of semi-streaming graph algorithms. Our algorithm uses an extension of a well studied approximation framework, namely the local-ratio technique, while previous algorithm used clever ideas which were specifically crafted for the problem and model.

As noted, a simple greedy algorithm gives a 2-approximation for MWM in the unweighted case. In the weighted case, a 2-approximation can be achieved by first sorting the edges and then adding them greedily, from the heaviest to the lightest. The local-ratio technique enables us to ignore some of the edges, while processing the other edges in an arbitrary order. In this work, we extend the local-ratio technique, in a way that allows us to discard all but $O(n \log n)$ of the edges.

A simple local-ratio algorithm for the MWM problem in the sequential model of computation goes roughly as follows: repeatedly select an edge with positive weight; reduce its weight from the edge itself and from all its neighboring edges; push the edge into a stack and continue to the next edge, as long as there is an edge with positive weight; finally, unwind the stack and add the edges greedily to the matching. This procedure results in a 2-approximation for the MWM problem. It can be extended to a (2α) -approximation, for $\alpha > 1$, if at each step we reduce the weight of the processed edge multiplied by α from its neighbors.

The challenge in translating this technique to the semi-streaming model is twofold. First, we have to reduce edge weights from edges that are yet to arrive. This is solved by saving, for each node, the total amount that should be reduced from each arriving edge containing this node, and reducing weight *retroactively* from incoming stream edges.

The second, more substantial challenge, is limiting the size of the stack, so it can comply with the $O(n \text{ polylog } n)$ space bound. It is not hard to come up with an execution of the above algorithm where all edges are eventually stored in the stack, which may take up to $\Omega(n^2 \text{ polylog } n)$ space. To overcome this problem, we *remove edges* from within the stack, *during* the execution of the algorithm. The traditional local-ratio technique was not designed to work under space limitations, and thus does not guarantee any approximation ratio if edges are removed from the stack. The crux of our approach is a variation of the local-ratio technique, which provides conditions under which an edge may be removed from the stack while incurring only a small loss in the approximation ratio.

Specifically, we show that if an edge in the stack is significantly lighter than its neighboring edge, and this neighboring edge is added to the stack, then removing the light edge has only a small effect on the total weight of the solution. In order to use this conclusion, we must first assure a steady increase in the edge weights around each node. This again requires some sophistication beyond the classical local-ratio approach for the problem.

We achieve this by increasing the weight an edge reduces from its neighborhood to be α -times its weight. This results in another deterioration in the approximation ratio, but has the benefit

of forcing the weights of edges in the stack to exhibit an exponential growth pattern. This, in turn, creates the conditions for our modified local-ratio theorem to show its strength, allowing us to keep the size of the stack within the model’s limits. Choosing parameters that carefully manage the tradeoff between the space and the approximation ratio, we achieve a $(2 + \epsilon)$ -approximation using $O(n \log^2 n)$ space. Ghaffari [Gha17] has recently improved the analysis of our algorithm, and showed a slight modification to the algorithm, which achieves the optimal $O(n \log n)$ -space bound.

Finally, we note that the basic structure of the local-ratio technique, namely processing the edges one by one in an *arbitrary* order and then performing some postprocessing, suits very naturally to the streaming environment. Combined with the machinery we develop here in order to follow the semi-streaming space constraints, we believe this technique can be applied to additional problems in the semi-streaming model and in similar computational models.

Related Work The study of graph algorithms in the semi-streaming model was initiated by Feigenbaum et al. [FKM⁺05], in order to tackle the problem of processing massive graphs whose edge set cannot be stored in memory. The need for algorithms for such massive graphs is evident, as they become increasingly common: graphs representing social networks, graphs for metabolic interactions used in computational biology and even the communication graph of the Internet, are only a few examples.

Feigenbaum et al. were also the first to study the MWM problem in the semi-streaming model. They show a 6-approximation algorithm for MWM, which maintains a matching by adding an edge to it and removing the edge’s neighbors only if the edge’s weights is twice the weight of the removed edges. This idea was later improved by McGregor [McG05] to achieve an approximation ratio of 5.828, by changing the threshold for inserting an edge to the matching. McGregor also presents a $(2 + \epsilon)$ -approximation algorithm for the problem, but with $O(\epsilon^3)$ passes on the input. By using the same ideas, while keeping deleted edges and reviving them later, Zelke [Zel12] achieves a 5.585-approximation algorithm.

A different approach was taken by Epstein et al. [ELMS11], who achieve a $(4.911 + \epsilon)$ -approximation algorithm. They use bucketing, i.e. separate the edges into $O(\log n)$ weight classes, find a matching in each bucket, and then find the final matching in the union of these matching. Crouch and Stubbs [CS14] achieve an approximation ratio of $(4 + \epsilon)$ using related ideas, but their algorithm use weight classes which are unbounded from above, and thus are not disjoint. Recently, Grigorescu et al. [GMZ16] improved the analysis of the last algorithm, to show it achieves an approximation ratio of 3.5.

The bucketing technique takes a heavy toll on the approximation factor, and Crouch and Stubbs [CS14] prove this technique cannot give an approximation ratio better than 3.5. To circumvent this bound, we use a different approximation framework, the local-ratio technique. To the best of our knowledge, this is the first use of this technique in the streaming environment.

A related problem is estimating size of the maximum matching in a graph [AKL17, Kap13, KKS14, GKK12] which is known to be related to matrix rank approximation. More general submodular-function matching problems in the semi-streaming model have been considered by Varadaraja and by Chakrabarti and Kale [Var11, CK14].

The problem of MWM was also considered in other streaming models, such as the MapReduce model [CS14, LMSV11], the sliding-window model [CS14, CMS13] and the turnstile stream model (allowing deletions as well as insertions) [Kon15, AKLY16, BS15, CCE⁺16]. Extending our technique to other computational models is a challenge yet to be addressed.

Structure of this Paper We formally define the MWM problem and the semi-streaming model of computation in Section 2. In Section 3 we introduce the local-ratio theorem, present a sequential 2-approximation local-ratio algorithm for MWM in the sequential model, and discuss our extensions to the theorem. In Section 4 we extend the 2-approximation algorithm to a more involved $(2 + \epsilon)$ -approximation algorithm for MWM and analyze its performance, and finally adapt the last algorithm to the semi-streaming model.

2 Preliminaries

Let $G = (V, E, w)$ be a simple graph with non-negative edge weights, $w : E \rightarrow \mathbb{R}_{\geq 0}$. Denote $n = |V|$ and $m = |E|$; the *neighboring edges* of e are $N(e) = \{e' \mid |e \cap e'| = 1\}$, and $N^+(e) = N(e) \cup \{e\}$. We usually assume edge weights can be represented by $O(\log n)$ bits, and discuss greater weights at the end of the paper.

Maximum Weighted Matching A *matching* in G is a set $M \subseteq E$ of edges such that no two edges share a node. A *maximum weight matching* (MWM) in G is a matching M of maximum weight: for every matching M' in G , we have $\sum_{e \in M} w(e) \geq \sum_{e \in M'} w(e)$.

We represent edge weights and matchings using vectors indexed by the edges. The weight function is identified with a vector w , where $w[e]$ is the weight of the edge e . A matching M is identified with its indicator vector x , defined as $x[e] = 1$ if $e \in M$, and $x[e] = 0$ otherwise. The weight of a matching x is the value of the inner product of x and w , denoted xw . A set of feasibility constraints on x is induced by the graph in a straightforward manner.

Approximation Algorithms A feasible matching x is said to be a p -*approximation* of a MWM in G , for a constant $p \geq 1$, if every matching x^* satisfies $x^*w \leq p \cdot xw$. An algorithm returning a p -approximation on every input graph is said to be a p -*approximation algorithm* for the MWM problem, and p is called the *approximation ratio* of the algorithm. Note that if $p' > p$ then a p -approximation algorithm is also a p' -approximation algorithm.

The Semi-Streaming Model In the semi-streaming model of computation, the input graph is given as a stream of edges. At each iteration, the algorithm receives an edge from the stream and processes it. As in sequential algorithms, we wish to compute some function of the input graph. But, we assume the number of edges in the graph is too large to fit in memory, so we limit the algorithm to use only $O(n \text{ polylog } n)$ space. Moreover, if processing an incoming edge takes long, we have to keep a queue of later incoming edges, which may result in exceeding the space limitations; thus, short processing time for an incoming edge is of high importance, while pre-processing and post-processing times are of less importance.

3 Approximating Maximum Weight Matching

In this section we present the *local-ratio* theorem for maximization problems [BBFR04], and use it to present a sequential 2-approximation algorithm for MWM. We then present extensions of this technique and use them in order to adjust the sequential local-ratio algorithm to the semi-streaming model, incurring only a small loss in the approximation ratio.

3.1 A Simple Local-Ratio Approximation Algorithm for MWM

The basic building blocks in local-ratio algorithms are the *weight reduction* steps. A weight reduction step on $G = (V, E, w)$ is done by defining two edge-weight functions on (V, E) , the *reduced graph* with weight vector w_r and the *residual graph* with weight vector \bar{w}_r , such that $w = w_r + \bar{w}_r$. We start with the local-ratio theorem for maximization problems [BBFR04, Theorem 9], which we restate here for completeness. Note that this theorem applies even if w_r takes negative values.

Theorem 1. *Let $w \in \mathbb{R}^m$ be a vector, and consider the problem of maximizing the product xw under a set of feasibility constraints. Let $w_r, \bar{w}_r \in \mathbb{R}^m$ be vectors such that $w = w_r + \bar{w}_r$. Let $x \in \mathbb{R}^m$ be a feasible solution that is a p -approximation with respect to w_r and with respect to \bar{w}_r . Then x is a p -approximation with respect to w as well.*

Proof. Let x^*, x_r^* and \bar{x}_r^* be maximum feasible solutions with respect to w, w_r and \bar{w}_r . Then

$$\begin{aligned} x^* w &= x^* w_r + x^* \bar{w}_r \\ &\leq x_r^* w_r + \bar{x}_r^* \bar{w}_r \\ &\leq p \cdot x w_r + p \cdot x \bar{w}_r \\ &= p \cdot x w, \end{aligned}$$

where the first inequality follows from the maximality of x_r^* and \bar{x}_r^* , and the second from the assumption that x is a p -approximation with respect to w_r and \bar{w}_r . \square

We apply weight reduction steps iteratively, while assuring that any p -approximate solution to w_r can be easily extended into a p -approximate solution to \bar{w}_r .

For the specific problem of MWM, a weight reduction step is done by picking an arbitrary edge $e \in E$ of positive weight and reducing the weight of e from every $e' \in N^+(e)$. This splits the weight vector w into two vectors, w_r and \bar{w}_r , as follows: $\bar{w}_r[e'] = w[e]$ for every $e' \in N^+(e)$, $\bar{w}_r[e] = 0$ for every other edge, and $w_r = w - \bar{w}_r$. Any 2-approximate solution for the reduced graph can be easily extended into a 2-approximate solution for the residual graph by making sure that at least some $e' \in N^+(e)$ is in the solution: if this is not the case, we can add e to the solution without violating the constraints. As $w_r[e] = 0$, adding e to the solution does not change the solution's value. Thus, we get a 2-approximate solution for both w_r and \bar{w}_r .

This simple technique is realized by Algorithm 1. First, it applies weight reduction steps iteratively using edges of positive reduced weight, splitting a weight function w_i into w_{i+1} (reduced) and \bar{w}_{i+1} (residual) and keeping the edge in a stack. When no edges with positive reduced weights remain, the algorithm unwinds the stack and adds the edges greedily to the matching. When we unwind the stack we maintain a set of interim solutions $\{x_i\}$. We use the local-ratio theorem to guarantee that every x_i is a 2-approximate solution for w_i . Finally returning x_1 which is a 2-approximate solution for the original problem.

We also note that this algorithm does not work in the semi-streaming model, as the stack can easily grow to contain $\Omega(n^2)$ edges.

3.2 Extending the Local-Ratio Technique

We now extend the approximation techniques used in Algorithm 1. This allows us to present another sequential approximation algorithm for MWM in the following section, with a worse approximation

Algorithm 1: `MWM-simple`(V, E, w). A simple 2-approximation algorithm for MWM

```

1  $S \leftarrow$  empty stack
2  $w_1 \leftarrow w$ ;  $i \leftarrow 1$ 
3 foreach  $e_i \in E$  s.t.  $w_i(e_i) \geq 0$  do
4    $S.push(e_i)$ 
5    $w_{i+1} \leftarrow w_i$ 
6   foreach  $e' \in N^+(e_i)$  do
7      $w_{i+1}[e'] \leftarrow w_i[e'] - w_i[e_i]$ 
7     /* Implicit:  $\bar{w}_{i+1}[e'] \leftarrow w_i[e_i]$  for every  $e' \in N^+(e_i)$  */
8    $i \leftarrow i + 1$ 
9  $k \leftarrow |S|$ 
10  $x_{k+1} \leftarrow \vec{0}$ 
11 for  $i \leftarrow k$  down to 1 do
12    $x_i \leftarrow x_{i+1}$ 
13    $e_i \leftarrow S[i]$ 
14   if  $\forall e \in N(e_i) : x_i[e] = 0$  then
15      $x_i[e_i] \leftarrow 1$ 
16 return  $x_1$ 

```

ratio of $2 + \epsilon$. However, from the new algorithm we derive the desired approximation algorithm for the semi-streaming model, with no further increase in the approximation ratio.

If instead of reducing exactly $w[e]$ from the neighboring edges of e , we reduce $w[e]$ or $\alpha w[e]$ from each such edge, for some $\alpha \geq 1$, we get a (2α) -approximation, as formalized in the next claim.

Lemma 2. *Let w, w_r and \bar{w}_r be weight functions and $e \in E$ an edge, such that*

$$\bar{w}_r[e'] = \begin{cases} w[e] & e' = e; \\ \alpha w[e] \text{ or } w[e] & e' \in N(e); \\ 0 & \text{otherwise,} \end{cases} \quad (1)$$

and $w_r = w - \bar{w}_r$; the choice between $w[e]$ and $\alpha w[e]$ can be arbitrary.

Let $x \in \{0, 1\}^m$ be a matching. If $x[e'] \neq 0$ for some $e' \in N^+(e)$, then x is a (2α) -approximate solution for \bar{w}_r .

Proof. Let x^* be any matching. The definition of \bar{w}_r guarantees that x^* contains at most two edges of non-zero weight in \bar{w}_r , each of weight at most $\alpha w[e]$, so $x^* \bar{w}_r \leq 2\alpha w[e]$. On the other hand, $x[e'] \neq 0$ for some $e' \in N^+(e)$, so $w[e] \leq x \bar{w}_r$. The claim follows. \square

Next, we note that if the optimal solution for the reduced graph is greater than the optimal solution for the residual graph by some multiplicative factor $p \geq 1$, then it is also a $(1 + 1/p)$ -approximation for the original graph. For large values of p , an approximate solution for the reduced graph gives roughly the same approximation ratio for the original graph, which allows us to ignore the residual graph. We formalize this in the next lemma.

Lemma 3. Let w, w_r and \bar{w}_r be weight functions satisfying $w = w_r + \bar{w}_r$ and $w_r[e] \leq w[e]$ for all $e \in E$. Let x_r be a β -approximate solution for w_r .

If $x_r w_r$ is at least p times larger than any matching in \bar{w}_r , then x_r is a $(\beta + 1/p)$ -approximate solution for w .

Proof. Let x^*, x_r^* and \bar{x}_r^* be matchings of maximum weights in w, w_r and \bar{w}_r respectively.

The assumptions imply $x_r^* w_r \leq \beta x_r w_r$ and $p \bar{x}_r^* \bar{w}_r \leq x_r w_r$, so

$$\begin{aligned} x^* w &= x^* w_r + x^* \bar{w}_r \\ &\leq x_r^* w_r + \bar{x}_r^* \bar{w}_r \\ &\leq \beta x_r w_r + (1/p) x_r w_r \\ &= (\beta + 1/p) x_r w_r \\ &\leq (\beta + 1/p) x_r w, \end{aligned}$$

where the last inequality follows from the fact that $w_r[e] \leq w[e]$ for all $e \in E$. \square

Let w_1 be a weight vector for the MWM problem, and consider an iterative splitting of w_i into w_{i+1} and \bar{w}_{i+1} for k times. The last lemma allowed us to ignore the residual graph once; we now extend it to allow the iterative omission of the residual graph.

Denote $\alpha = \sqrt{1 + \epsilon/2}$, $\gamma = n^2 / \ln(\alpha)$, and $\beta_i = 2\alpha(1 + 1/\gamma)^{k+1-i}$ for all i .

Lemma 4. Let $G = (V, E, w_1)$ a graph, and w_2, \dots, w_{k+1} and $\bar{w}_2, \dots, \bar{w}_{k+1}$ sequences of reduced and residual weight functions for (V, E) , respectively.

Assume that we generate a sequence of solutions x_{k+1}, \dots, x_1 , such that x_{k+1} is an optimal solution for w_{k+1} , and that for $1 \leq i \leq k$, if x_{i+1} is a β_{i+1} -approximate solution for w_{i+1} then x_i has the following properties:

1. x_i is a β_{i+1} -approximate solution for w_{i+1} .
2. At least one of the following holds:
 - (a) x_i is a β_{i+1} -approximate solution for \bar{w}_{i+1} ; or
 - (b) $x_i w_{i+1} \geq (\gamma / \beta_{i+1}) x^* \bar{w}_{i+1}$ for every solution x^* .

Then, x_1 is a β_1 -approximate solution for w_1 .

Proof. We prove, by induction on i ranging from $k+1$ down to 1, that x_i is a β_i -approximate solution for w_i .

The base, $i = k+1$, is trivial by the assumption on x_{k+1} .

Assume the claim is true for x_{i+1} , then condition 1 holds for x_i . If condition 2(a) holds, then by condition 1 and the local-ratio theorem (Theorem 1), x_i is a β_{i+1} -approximate solution for w_i . Because $\beta_i > \beta_{i+1}$, x_i is also a β_i -approximate solution for w_i . If condition 2(b) holds, then from condition 1 and Lemma 3 we deduce that x_i is a $(\beta_{i+1} + \beta_{i+1}/\gamma)$ -approximate solution for w_i . The definition of β_i yields:

$$\begin{aligned} \beta_{i+1} + \beta_{i+1}/\gamma &= (1 + 1/\gamma) \cdot 2\alpha(1 + 1/\gamma)^{k+1-(i+1)} \\ &= 2\alpha(1 + 1/\gamma)^{k+1-i} = \beta_i. \end{aligned}$$

Specifically, x_1 is a β_1 -approximate solution for w_1 , and the proof is complete. \square

4 A Semi-Streaming Algorithm

We present a $(2 + \epsilon)$ -approximation algorithm for the MWM problem, using our extensions to the local-ratio technique. This algorithm could be used in the semi-streaming model if no space constraints applied. We give a detailed analyses of this algorithm, and then present a lightweight variant of it which obeys the space constraints.

The new algorithm is similar to Algorithm 1: it performs a series of weight reduction steps generating a series of reduced weight functions $\{w_i\}$, and then constructs a series of approximate solutions $\{x_i\}$. To prove the desired approximation ratio is achieved, we use Lemma 4 as a substitute for the local-ratio theorem.

We start by presenting the challenges posed by the semi-streaming model, and the ways in which the new algorithm deals with them.

Retroactive weight reduction The sequential algorithm constructs w_{i+1} from w_i using an edge e_i , by reducing $w_i(e_i)$ from the weight of every $e' \in N^+(e_i)$. This cannot be done directly in the semi-streaming model, as some edges of $N^+(e_i)$ may not have arrived yet. Instead, the algorithm keeps a variable $\phi_i(v) = \sum_{j=1}^i w_j[e_j]$ for every node $v \in V$. When a new edge $e = (u, v)$ arrives, its reduced weight is computed *retroactively* by reducing $\phi_{i-1}(u)$ and $\phi_{i-1}(v)$ from its original weight.

Removing edges from the stack In the sequential algorithm, the stack may grow to hold all of the graph edges. Lemma 4 presents conditions under which an approximate solution for w_{i+1} is also an approximate solution for w_i . When these conditions hold, we may remove the edge e_i from the stack, which we use in order to make sure the stack's size does not exceed $O(n \text{ polylog } n)$ edges. The new algorithm does not discard edges from the stack, but only replaces them with the \perp symbol; this is only done for the sake of analysis.

Assuring edge-weight growth In order to make sure edges are removed from the stack, we force a small but consistent growth in the edge weights around each node. Roughly speaking, the edge weights grow exponentially by a multiplicative α factor; after a logarithmic number of new edges considered, the weights grow large enough to allow the algorithm to neglect the older edges and remove them from the stack.

4.1 Algorithm MWM-seq

Algorithm **MWM-seq** (Algorithm 2) has two phases: in the first phase, it iterates over the edges and pushes chosen edges into a stack. In the second phase, the edges are popped out of the stack and added greedily to the matching.

The algorithm begins with an edge-weight function w_1 , given as input. For each node u , the algorithm explicitly maintains a non-negative weight function $\phi_i(u)$, which is used to filter edges (Line 6): an edge $e = (u, u')$ processed at iteration i is *light* if $w_1[e] \leq \alpha(\phi_{i-1}(u) + \phi_{i-1}(u'))$, and *heavy* otherwise. Iterations are defined by the heavy edges. When the first heavy edge in iteration $i - 1$ arrives, it is denoted e_i , and iteration i begins. Thus, all heavy edges are eventually denoted with sub-indexes (e_i), while the light edges are left un-tagged (e).

When an edge $e = (u, v)$ is processed in iteration i , the algorithm performs all weight reduction steps on e retroactively using $\phi_{i-1}(v)$ and $\phi_{i-1}(u)$, to set the value of $w_i[e]$. It decides between

reducing $\phi_{i-1}(v) + \phi_{i-1}(u)$ or $\alpha(\phi_{i-1}(v) + \phi_{i-1}(u))$ from the weight of e , in a way that guarantees an exponential growth of ϕ , implying a bound on the size of the stack.

For every node u , we hold a queue $E_i(u)$. This is a list of heavy edges containing the node u currently present in the stack. Upon seeing a heavy edge, $e_i = (u, v)$, we perform a weight reduction step: $\phi_{i-1}(u)$ and $\phi_{i-1}(v)$ are increased by $w_i[e_i]$, and e_i is pushed into the stack. We also enqueue e_i in $E_i(u), E_i(v)$. If the size of $E_i(u)$ exceeds a certain bound, we dequeue an edge from $E_i(u)$, and remove it from the stack. For the sake of analysis, we insert \perp in its place in the stack.

In the second phase, the algorithm unwinds the stack, adding edges greedily to the matching while ignoring the \perp symbol. The usage of the \perp symbol is replaced by deletion of the relevant edge in the semi-streaming algorithm, presented in the next subsection.

We start the analyses of Algorithm **MWM-seq** by proving that the node-weight functions $\phi_i(u)$ grow exponentially. In the algorithm, the variable $c_i(u)$ counts the heavy edges containing u that arrived until iteration i . Its value is not used in the algorithm itself; we only use it in the proof, to bound from below the growth $\phi(u)$. In various places in the proof we consider the expression $c_i(u) - c_j(u)$. This is the amount of heavy edges added to u from iteration j until iteration i . We will eventually show that the reduced weights of heavy edges exhibit a growth pattern exponential in $c_i(u) - c_j(u)$.

Lemma 5. *For every $u \in V$ and $i \geq j$, it holds that $\phi_i(u) \geq \alpha^{c_i(u)-c_j(u)}\phi_j(u)$.*

Proof. We fix j , and prove by induction on i , where $i \geq j$. The base case, $i = j$, is trivial.

For $i > j$, we consider two cases: if $u \notin e_i$ then $c_i(u) = c_{i-1}(u)$, so $\phi_i(u) = \phi_{i-1}(u) \geq \alpha^{c_{i-1}(u)-c_j(u)} = \alpha^{c_i(u)-c_j(u)}$ by the induction hypothesis.

Otherwise, $e_i = (u, u')$ for some $u' \in V$, so $c_i(u) = c_{i-1}(u) + 1$, and

$$\begin{aligned} \phi_i(u) &= \phi_{i-1}(u) + w_i[e_i] \\ &\geq \phi_{i-1}(u) + (\alpha - 1)(\phi_{i-1}(u) + \phi_{i-1}(u')) \\ &\geq \phi_{i-1}(u) + (\alpha - 1)\phi_{i-1}(u) \\ &= \alpha\phi_{i-1}(u) \\ &\geq \alpha \cdot \alpha^{c_{i-1}(u)-c_j(u)}\phi_j(u) \\ &= \alpha^{c_i(u)-c_j(u)}\phi_j(u) \end{aligned}$$

The first equality is due to Line 15, the first inequality follows Lines 6 and 10, and the last two transitions use the induction hypothesis and the fact that $c_i(u) = c_{i-1}(u) + 1$. \square

Consider the sequences of reduced and residual edge-weight functions, w_2, \dots, w_{k+1} and $\bar{w}_2, \dots, \bar{w}_{k+1}$, induced by the algorithm. While $w_i[e_i]$ is defined explicitly in the algorithm, the other values are only set after the first phase ends, and so does k , the length of the sequences.

The weight functions are defined inductively as follows. We formally define $w_1 = w$, where w is the function given as input. The edge e_i is used to split the weight function w_i into w_{i+1} and \bar{w}_{i+1} , the latter defined by

$$\bar{w}_{i+1}[e'] = \begin{cases} w_i[e_i] & e' = e_i; \\ w_i[e_i] & e' \in N(e_i) \text{ and } e' \text{ is heavy}; \\ \alpha w_i[e_i] & e' \in N(e_i) \text{ and } e' \text{ is light}; \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

Algorithm 2: MWM-seq(V, E, w). A sequential approximation algorithm for MWM

```

1  $S \leftarrow$  empty stack
2  $w_1 \leftarrow w$ ;  $\phi_0 \leftarrow \vec{0}$ ;  $c_0 \leftarrow \vec{0}$ 
   /*  $c_i$  is only used for the proof */
3  $\forall v \in V : E_0(v) \leftarrow$  empty queue
4  $i \leftarrow 1$ 
5 foreach  $e = (u, u') \in E$  do
6   if  $w_1[e] \leq \alpha(\phi_{i-1}(u) + \phi_{i-1}(u'))$  then
7     continue
   /* Implicit:  $\bar{w}_{j+1}[e] \leftarrow \alpha w_j[e_j]$  for every  $e_j \in N(e)$  */
8    $e_i \leftarrow e$ 
9    $S.\text{push}(e_i)$ 
10   $w_i[e_i] \leftarrow w_1[e_i] - (\phi_{i-1}(u) + \phi_{i-1}(u'))$ 
   /* Implicit:  $\bar{w}_{j+1}[e_i] \leftarrow w_j[e_j]$  for every  $e_j \in N^+(e_i)$  */
11   $\phi_i \leftarrow \phi_{i-1}$ ;  $E_i \leftarrow E_{i-1}$ ;  $c_i \leftarrow c_{i-1}$ 
12  foreach  $v \in e$  do
13     $c_i(v) \leftarrow c_i(v) + 1$ 
14     $E_i(v).\text{enqueue}(e_i)$ 
15     $\phi_i(v) \leftarrow \phi_{i-1}(v) + w_i[e_i]$ 
16    if  $(\alpha - 1)\alpha^{|E_i(v)|-2} > 2\alpha\gamma$  then
17       $e_j \leftarrow E_i(v).\text{dequeue}()$ 
18       $S[j] \leftarrow \perp$ 
19   $i \leftarrow i + 1$ 
20  $k \leftarrow |S|$ 
21  $x_{k+1} \leftarrow \vec{0}$ 
22 for  $i \leftarrow k$  down to 1 do
23    $x_i \leftarrow x_{i+1}$ 
24    $e_i \leftarrow S[i]$ 
25   if  $e_i = \perp$  then
26     continue
27   if  $\forall e \in N(e_i) : x_i[e] = 0$  then
28      $x_i[e_i] \leftarrow 1$ 
29 return  $x_1$ 

```

and the former by $w_{i+1} = w_i - \bar{w}_{i+1}$. The length k is the number of heavy edges encountered in the first phase. Note that \bar{w} is non-negative, so $w_i[e]$ is a non-increasing function of i , for any fixed edge e .

The next lemma focuses on a node u and the heavy edges adjacent to it. It asserts that for $i > j$, the reduced weight at iteration $j + 1$ of a heavy edge e_i grows exponentially with respect to $w_j[e_j]$.

Lemma 6. *Let $e_i, e_j \in E$ such that $i > j$ and $e_i \cap e_j = \{u\}$. Then $w_{j+1}[e_i] > (\alpha - 1)\alpha^{c_i(u) - c_j(u) - 1}w_j[e_j]$.*

Proof. The lemma follows by a simple computation. As $w_i(e)$ is a non-increasing

$$\begin{aligned}
w_{j+1}(e_i) &\geq w_i(e_i) \\
&\geq (\alpha - 1)\phi_{i-1}(u) && \text{(Lines 6 and 10)} \\
&\geq (\alpha - 1)\alpha^{c_{i-1}(u)-c_j(u)}\phi_j(u) && \text{(Lemma 5)} \\
&\geq (\alpha - 1)\alpha^{c_{i-1}(u)-c_j(u)}w_j[e_j] && \text{(Line 15)} \\
&= (\alpha - 1)\alpha^{c_i(u)-c_j(u)-1}w_j[e_j] && (u \in e_i \text{ implies } c_{i-1}(u) = c_i(u) - 1)
\end{aligned}$$

as desired. \square

In the second loop of the algorithm, the edges are taken out of the stack and a solution is greedily constructed. The algorithm's approximation ratio is the approximation ratio of the solution x_1 on the original weight function w_1 . To bound this quantity, we prove by induction that every x_i is a β_i -approximate solution for w_i . We break our analysis into cases, for which we need the next three lemmas.

Lemma 7. *If x_{i+1} is a β_{i+1} -approximate solution for w_{i+1} and the condition in Line 25 holds for e_i , then x_i is a β_i -approximate solution for w_i .*

Proof. Since the condition in Line 25 holds, we have $x_i = x_{i+1}$. This immediately guarantees that x_i is a feasible solution and that condition 1 of Lemma 4 holds. We show that condition 2(b) of Lemma 4 holds as well.

Let $e_i = (u, u')$. Because the condition in Line 25 holds, we know that in some iteration i' , $i' > i$, the condition in Line 16 held as well. That is, $e_{i'}$ was enqueued into $E_{i'}(v)$, the condition $(\alpha - 1)\alpha^{|E_{i'}(v)|-2} > 2\alpha\gamma$ held, and e_i was then dequeued from $E_{i'}(v)$. Every enqueue operation to $E_i(v)$ is accompanied by an increase of $c_i(v)$ by 1, so when the condition in Line 16 was checked, e_i and $e_{i'}$ were the first and last elements in $E_{i'}(v)$, and the size of $E_{i'}(v)$ was exactly $c_{i'}(v) - c_i(v) + 1$. Thus, $(\alpha - 1)\alpha^{c_{i'}(v)-c_i(v)-1} \geq 2\alpha\gamma$.

Using this inequality and Lemma 6, we have

$$w_{i+1}[e_{i'}] \geq (\alpha - 1)\alpha^{c_{i'}(v)-c_i(v)-1}w_i[e_i] \geq 2\alpha\gamma w_i[e_i].$$

Hence, the single edge $e_{i'}$ is a matching of weight at least $2\alpha\gamma w_i[e_i]$ in w_{i+1} . As x_{i+1} is a β_{i+1} -approximate solution for w_{i+1} , we have $\beta_{i+1}x_{i+1}w_{i+1} \geq 2\alpha\gamma w_i[e_i]$.

The definition of \bar{w}_{i+1} guarantees it has the following structure:

$$\bar{w}_{i+1}[e'] = \begin{cases} w_i[e_i] & e' = e_i; \\ \alpha w_i[e_i] \text{ or } w_i[e_i] & e' \in N(e_i); \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

Thus, any solution x^* for \bar{w}_{i+1} contains at most two edges, of weight at most $\alpha w_i[e_i]$, i.e. $2\alpha w_i[e_i] \geq x^*\bar{w}_{i+1}$. The last two inequalities guarantee any solution x^* satisfies

$$(\beta_{i+1}/\gamma)x_{i+1}w_{i+1} \geq 2\alpha w_i[e_i] \geq x^*\bar{w}_{i+1}$$

so $x_i w_{i+1} = x_{i+1} w_{i+1} \geq (\gamma/\beta_{i+1})x^*\bar{w}_{i+1}$, and condition 2(b) of Lemma 4 holds. \square

Lemma 8. *If x_{i+1} is a β_{i+1} -approximation for w_{i+1} and the condition on line 25 does not hold for e_i , then x_i is a β_i -approximation for w_i .*

Proof. If the condition on Line 27 holds, then x_i is derived from x_{i+1} by adding e_i to x_{i+1} . The condition in this line, together with the assumption that x_{i+1} is a matching, guarantee that x_i is a matching. Since $\bar{w}_{i+1}[e_i] = w_i[e_i]$ and $w_{i+1} = w_i - \bar{w}_{i+1}$, we have $w_{i+1}[e_i] = 0$. Hence, $x_i w_{i+1} = x_{i+1} w_{i+1}$, so x_i is also a β_{i+1} -approximate solution for w_{i+1} and condition 1 of Lemma 4 holds. By Lemma 2, x_i is a (2α) -approximate solution for \bar{w}_{i+1} , and because $2\alpha \leq \beta_{i+1}$ it is also a β_{i+1} -approximate solution to \bar{w}_{i+1} and condition 2(a) of Lemma 4 holds.

Finally, if the condition on line Line 27 does not hold, we set $x_i = x_{i+1}$. Then x_i is a feasible matching satisfying condition 1 of Lemma 4. The condition in Line 27 does not hold, so $x_{i+1}[e'] \neq 0$ for some $e' \in N^+[e_i]$, and Lemma 2 promises x_i is a (2α) -approximation for \bar{w}_{i+1} . As before, $2\alpha \leq \beta_{i+1}$ proves condition 2(a) of Lemma 4 holds. \square

The next lemma asserts that when the first phase ends, none of the reduced edge weights is positive.

Lemma 9. *At the end of the first phase, $w_{k+1}[e] \leq 0$ for all $e \in E$.*

Proof. Consider an edge e . If $e = e_i$ is heavy then $\bar{w}_{i+1}[e_i] = w_i[e_i]$ and $w_{i+1} = w_i - \bar{w}_{i+1}$ imply $w_{i+1}[e_i] = 0$. The monotonicity of $w_i[e]$ completes the proof.

If $e = (u, u')$ is a light edge considered in iteration i , then $w_1[e] \leq \alpha(\phi_{i-1}(u) + \phi_{i-1}(u'))$. Line 15 guarantees

$$\phi_{i-1}(u) = \sum_{\substack{e_j \mid u \in e_j \\ j \leq i-1}} w_j[e_j],$$

and a similar claim holds for u' . On the other hand, $w_{j+1} = w_j - \bar{w}_{j+1}$ and $\bar{w}_{j+1}[e] = \alpha w_j[e_j]$ for all $e_j \in N(e)$. Hence $w_{j+1}[e] = w_j[e] - \alpha w_j[e_j]$, and a simple induction implies

$$w_i[e] = w_1[e] - \alpha \sum_{\substack{e_j \in N(e) \\ j \leq i-1}} w_j[e_j].$$

The last two equalities, together with the definition of $N(e)$, imply $w_i[e] = w_1[e] - \alpha(\phi_{i-1}(u) + \phi_{i-1}(u'))$. The inequality $w_1[e] \leq \alpha(\phi_{i-1}(u) + \phi_{i-1}(u'))$ implies $w_i[e] \leq 0$ for all $e \in E$, and the monotonicity of $w_i[e]$ completes the proof. \square

We are now ready to prove the main theorem of this section.

Theorem 10. *Algorithm **MWM-seq** returns a $(2 + \epsilon)$ -approximation for the MWM problem.*

Proof. By Lemma 9, the first loop ends when $w_{k+1} \leq \vec{0}$, so $x_{k+1} = \vec{0}$ is indeed an optimal solution for w_{k+1} .

Assume x_{i+1} is a β_{i+1} -approximate solution for w_{i+1} . From Lemmas 7 and 8 we conclude that in all cases the conditions of Lemma 4 hold, so x_1 is a β_1 -approximate solution for $w = w_1$.

Substitute $\beta_1 = 2\alpha(1 + 1/\gamma)^k$, $\alpha = \sqrt{1 + \epsilon/2}$ and $\gamma = n^2/\ln(\alpha)$, and note $k \leq m \leq n^2$, to get

$$\begin{aligned} \beta_1 &\leq 2\alpha(1 + 1/\gamma)^{n^2} \\ &= 2\alpha \left(1 + (\ln \alpha)/n^2\right)^{n^2} \\ &\leq 2\alpha e^{\ln \alpha} = 2 + \epsilon. \end{aligned}$$

Algorithm 3: $\text{MWM-semi}(V, E, w)$. A Semi-Streaming approximation algorithm for MWM

```

1  $S \leftarrow$  empty stack
2  $\phi \leftarrow \vec{0}$ 
3  $\forall v \in V : E(v) \leftarrow$  empty queue
4 foreach  $e = (u, u') \in E$  do
5   if  $w[e] \leq \alpha(\phi(u) + \phi(u'))$  then
6      $\perp$  continue
7    $S.\text{push}(e)$ 
8    $w'[e] \leftarrow w[e] - (\phi(u) + \phi(u'))$ 
9   foreach  $v \in e$  do
10     $E(v).\text{enqueue}(e)$ 
11     $\phi(v) \leftarrow \phi(v) + w'[e]$ 
12    if  $(\alpha - 1)\alpha^{|E(v)|-2} > 2\alpha\gamma$  then
13       $e' \leftarrow E(v).\text{dequeue}()$ 
14      remove  $e'$  from  $S$ 
15  $M \leftarrow \emptyset$ 
16 while  $S \neq \emptyset$  do
17    $e \leftarrow S.\text{pop}()$ 
18   if  $M \cap N(e) = \emptyset$  then
19      $M \leftarrow M \cup \{e\}$ 
20 return  $M$ 

```

The desired approximation ratio is achieved. \square

4.2 Implementing Algorithm MWM-seq in the Semi-Streaming Model

In the previous section we showed that Algorithm MWM-seq computes a $(2 + \epsilon)$ -approximation for MWM. In the semi-streaming model, we must obey space constraints in addition to maintaining a good approximation ratio. In the presentation of the sequential algorithm we ignored the space constraints: we did not remove edges from the stack, and we represented the temporary solutions as the vectors x_i of size $\Theta(n^2)$.

In order to follow the space constraints, we replace any insertion of \perp into the stack by a removal of the relevant edge, and the vectors x_i by a single set containing the current matching. For the sake of completeness, we present Algorithm MWM-semi (Algorithm 3), an implementation of Algorithm MWM-seq in the semi-streaming model. The correctness of Algorithm MWM-semi is derived directly from the correctness of Algorithm MWM-seq , so we only need to prove it obeys the space constraints.

After omitting notations and auxiliary variables from Algorithm MWM-seq , we are only left with three types of data structures in Algorithm MWM-semi : M is the matching constructed, S is the stack and $E(v)$ is a queue of edges from S that contain node v . Every edge (u, v) that is added to S is also added to $E(v)$ and $E(u)$. When (u, v) is removed from S , it is also removed from $E(u)$ and $E(v)$, implying $|S| \leq \sum_v |E(v)|$. The next lemma bounds the size of $E(v)$ for every v .

Lemma 11. *During the execution of Algorithm **MWM-semi**, $|E(v)| = O\left(\frac{\log n + \log(1/\epsilon)}{\epsilon}\right)$ for each $v \in V$.*

Proof. After each iteration of the loop in Lines 4–14, we have $(\alpha - 1)\alpha^{|E(v)|-2} \leq 2\alpha\gamma$ for each $v \in V$: this is true at the beginning; $E(v)$ can grow only by 1 at each iteration; and whenever the inequality does not hold, an edge is removed from $E(v)$.

From the above inequality, we derive an asymptotic bound for $|E(v)|$.

$$\begin{aligned} |E(v)| &= O\left(\log_{\alpha} \frac{\alpha\gamma}{\alpha-1}\right) \\ &= O\left(\frac{\log \gamma + \log \alpha - \log(\alpha-1)}{\log \alpha}\right) \\ &= O\left(\frac{\log \gamma - \log(\alpha-1)}{\log \alpha}\right) \\ &= O\left(\frac{\log \gamma - \log((\alpha^2-1)/(\alpha+1))}{\log \alpha}\right) \\ &= O\left(\frac{\log \gamma - \log(\alpha^2-1)}{\log \alpha}\right). \end{aligned}$$

Plugging in $\alpha = \sqrt{1 + \epsilon/2}$ and $\gamma = n^2/\ln(\alpha)$ we get

$$\begin{aligned} |E(v)| &= O\left(\frac{\log \gamma - \log(\alpha^2-1)}{\log \alpha}\right) \\ &= O\left(\frac{\log(n^2/\ln \alpha) + \log(1/\epsilon)}{\log(1+\epsilon)}\right) \\ &= O\left(\frac{\log n + \log(1/\epsilon)}{\log(1+\epsilon)}\right) \\ &= O\left(\frac{\log n + \log(1/\epsilon)}{\epsilon}\right). \end{aligned}$$

where the last step uses the inequality $\ln(1+x) \geq x/(1+x)$ for $x > -1$. □

From Lemma 11 we conclude that for a constant ϵ , Algorithm **MWM-semi** maintains at most $O(n \log n)$ edges, each represented by $O(\log n)$ bits. Thus, the total space used is $O(n \log^2 n)$. Our algorithm requires $O(1)$ time to process a new edge arriving from the stream, and finally we execute a post-processing step which requires $O(n \log n)$ time. We arrive at the main theorem of this section:

Theorem 12. *There exists an algorithm in the semi-streaming model computing a $(2+\epsilon)$ -approximation for MWM, using $O(\epsilon^{-1}n \log n \cdot (\log n + \log(1/\epsilon)))$ space and having an $O(1)$ processing time.*

In our analysis we assume that the weights of edges can be represented using $O(\log n)$ bits. If this is not the case, and the weights are bounded by some W , our algorithm requires $O(n(\log^2 n + \log W))$ space, as we keep a sum of weights for every node. For every W that can be represented using a polylogarithmic number of bits, this is still $O(n \text{ polylog } n)$ space.

Acknowledgments We thank Keren Censor-Hillel and Seri Khoury for helpful discussions, Seffi Naor for useful comments on the presentation, and the anonymous referees of SODA 2017 for their comments.

References

- [AKL17] Sepehr Assadi, Sanjeev Khanna, and Yang Li. On estimating maximum matching size in graph streams. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1723–1742, 2017.
- [AKLY16] Sepehr Assadi, Sanjeev Khanna, Yang Li, and Grigory Yaroslavtsev. Maximum matchings in dynamic graph streams and the simultaneous communication model. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1345–1364, 2016.
- [BBF⁺01] Amotz Bar-Noy, Reuven Bar-Yehuda, Ari Freund, Joseph Naor, and Baruch Schieber. A unified approach to approximating resource allocation and scheduling. *J. ACM*, 48(5):1069–1090, 2001.
- [BBFR04] Reuven Bar-Yehuda, Keren Bendel, Ari Freund, and Dror Rawitz. Local ratio: A unified framework for approximation algorithms. In *memoriam: Shimon Even 1935-2004. ACM Comput. Surv.*, 36(4):422–463, 2004.
- [BS15] Marc Bury and Chris Schwiegelshohn. Sublinear estimation of weighted matchings in dynamic data streams. In *Algorithms - ESA 2015 - 23rd Annual European Symposium*, pages 263–274, 2015.
- [BYE85] Reuven Bar-Yehuda and Shimon Even. A local-ratio theorem for approximating the weighted vertex cover problem. *North-Holland Mathematics Studies*, 109:27–45, 1985.
- [CCE⁺16] Rajesh Chitnis, Graham Cormode, Hossein Esfandiari, MohammadTaghi Hajiaghayi, Andrew McGregor, Morteza Monemizadeh, and Sofya Vortnikova. Kernelization via sampling with applications to finding matchings and related problems in dynamic graph streams. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1326–1344, 2016.
- [CK14] Amit Chakrabarti and Sagar Kale. Submodular maximization meets streaming: Matchings, matroids, and more. In *Integer Programming and Combinatorial Optimization - 17th International Conference, (IPCO)*, pages 210–221, 2014.
- [CMS13] Michael S. Crouch, Andrew McGregor, and Daniel Stubbs. Dynamic graphs in the sliding-window model. In *Algorithms - ESA 2013 - 21st Annual European Symposium*, pages 337–348, 2013.
- [CS14] Michael Crouch and Daniel S. Stubbs. Improved streaming algorithms for weighted matching, via unweighted matching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 96–104, 2014.
- [Edm65] Jack Edmonds. Paths, trees, and flowers. *Canad. J. Math.*, 17:449–467, 1965.

- [ELMS11] Leah Epstein, Asaf Levin, Julián Mestre, and Danny Segev. Improved approximation guarantees for weighted matching in the semi-streaming model. *SIAM J. Discrete Math.*, 25(3):1251–1265, 2011.
- [FKM⁺05] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theor. Comput. Sci.*, 348(2-3):207–216, 2005.
- [Gha17] Mohsen Ghaffari. Space-optimal semi-streaming for $(2+\epsilon)$ -approximate matching. *CoRR*, abs/1701.03730, 2017.
- [GKK12] Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 468–485, 2012.
- [GMZ16] Elena Grigorescu, Morteza Monemizadeh, and Samson Zhou. Streaming weighted matchings: Optimal meets greedy. *CoRR*, abs/1608.01487, 2016.
- [Kap13] Michael Kapralov. Better bounds for matchings in the streaming model. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1679–1697, 2013.
- [KKS14] Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Approximating matching size from random streams. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 734–751, 2014.
- [Kon15] Christian Konrad. Maximum matching in turnstile streams. In *Algorithms - ESA 2015 - 23rd Annual European Symposium*, pages 840–852, 2015.
- [LMSV11] Silvio Lattanzi, Benjamin Moseley, Siddharth Suri, and Sergei Vassilvitskii. Filtering: a method for solving graph problems in mapreduce. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 85–94, 2011.
- [McG05] Andrew McGregor. Finding graph matchings in data streams. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 170–181, 2005.
- [MV80] S. Micali and V. V. Vazirani. An $O(\sqrt{|V|} \cdot |E|)$ algorithm for finding maximum matching in general graphs. In *Foundations of Computer Science, 1980*, pages 17–27, Oct 1980.
- [PS17] Ami Paz and Gregory Schwartzman. A $(2 + \epsilon)$ -approximation for maximum weight matching in the semi-streaming model. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2153–2161, 2017.
- [Var11] Ashwinkumar Badanidiyuru Varadaraja. Buyback problem - approximate matroid intersection with cancellation costs. In *Automata, Languages and Programming - 38th International Colloquium (ICALP)*, pages 379–390, 2011.
- [Zel12] Mariano Zelke. Weighted matching in the semi-streaming model. *Algorithmica*, 62(1-2):1–20, 2012.